Computer Communications 111 (2017) 142–152

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Length Shuffle: Achieving high performance and flexibility for data center networks design

Deshun Li, Yanming Shen*, Keqiu Li

School of Computer Science and Technology, Dalian University of Technology, No. 2 Linggong Road, Dalian 116024, China

ARTICLE INFO

Article history:
Received 27 December 2016
Revised 23 June 2017
Accepted 1 August 2017
Available online 8 August 2017

Keywords:
Data center network
High performance
Flexibility

ABSTRACT

Scale-out data center requires the underlying network to provide high performance and design flexibility on commodity switches. Current rigid architectures can provide high performance, but they hardly support network design flexibility. Flexible proposals address the issue of elasticity in the rigid networks. However, the introduction of random connection cannot provide guaranteed network performance. In this paper, we proposed Length Shuffle connection to address the performance issue in flexible data center networks. The proposed approach follows a greedy principle that gradually connects the furthest two switches with their available ports. Length Shuffle connection can reduce the number of long paths significantly, which results in a small average length of routing path. The proposed connection can work with random connection and regular connection in flexible network construction, and does not interfere the original routing protocols. Numerical analysis shows the nice properties of Length Shuffle connection in various dimensions. The extensive experimental results demonstrate the advantages of Length Shuffle connection in data transmission delay and aggregate throughput under different traffic patterns.

1. Introduction

Data center networks (DCNs) serve as an important infrastructure to provide public services, such as email, web search and online retail, as well as infrastructural services, such as cloud computing and big data processing [7,13,14,33,34,43,46]. Over the past decade, researchers have focused on tackling challenges of typical tree-based networks, and the following issues with respect to high performance and design flexibility in data center networks design [3,15,18,19,26,36,38,40]:

1. High performance. Many applications running in data center are data-intensive and time-sensitive. Data-intensive application requires high bandwidth to transport large chunk of data among servers, such as MapReduce [14], Hadoop [7]. Time-sensitive application transfers data to destination within a deadline time, such as web search, online retail, and advertisement. A small routing length will result in a short transmission delay, which can provide more bandwidth for applications and improve network throughput.
2. Design flexibility. Design flexibility has three aspects, arbitrary-sized network, incremental deployment and switches with different ports count. The requirement of arbitrary-sized network and incremental deployment is a practical strategy to reduce up-front expenses and respond to increasing applications [2,41]. Due to asymmetry of server distribution or switch heterogeneity, the inconsistent number of inter-switch ports in scale-out network connection should be taken into consideration.

During the past decade, a variety of network constructions have been proposed to meet the above-mentioned aspects, which fall into rigid and flexible topologies based on the structural characteristics The rigid networks is determined by the number of ports in a switch, which permits network scale with a very coarse parameter. For example, the multi-rooted tree based topologies can provide high performance on homogeneous switches [3,15,36], while its structure is completely formed by the number of ports in a switch, and it hardly supports incremental deployment. Several flexible solutions have been proposed to support on-demand network scale and incremental deployment [22,37,38,44,45]. For example, Jellyfish [38] introduces random connection to construct it's network, however, the resulting network cannot always provide guaranteed high network performance because of ruleless connection. Notice that the construction of network is a short-term investment, while the resulting networks provide a long-term and regular service. Therefore, it is cost-effective to spend more effort in construction to improve network performance with given equipments.

In this paper, we proposed Length Shuffle connection, which follows a greedy principle that connects two possible furthest

* Corresponding author.
E-mail addresses: lideshunlily@qq.com (D. Li), shen@dlut.edu.cn (Y. Shen).

http://dx.doi.org/10.1016/j.comcom.2017.08.001
0140-3664/

nodes gradually in each interconnection. Length Shuffle connection can be deployed on commodity top-of-rack (ToR) switches to construct flexible data center networks, which will reduce the number of long paths significantly and result in a small average length of routing path. The proposed connection can be deployed with random and regular connection in flexible network construction, which does not interfere the original routing protocols. The incremental deployment of Length Shuffle is also discussed in this paper. Length Shuffle connection demonstrates a high adaptability in various aspects, such as switch heterogeneity, hybrid deployment, incremental deployment, and fault tolerance. The numerical experiment shows that Length Shuffle connection provides a better performance than random connection. For example, the average path length in random connection is 2%∼5% larger than that of Length Shuffle connection on identical equipments. We conduct evaluations on the hybrid network of random and Length Shuffle connection under different traffic patterns. The results show that Length Shuffle connection can significantly reduce data transmission delay and improve throughput in heavy random and incast traffic patterns.

The rest of the paper is organized as follows. Section 2 introduces the related work and Section 3 presents the Length Shuffle connection. Sections 4 and 5 describes the deployment and measurement of the proposed connection. Section6 gives evaluation and Section 7 concludes our work.

2. Related work

The scale-out data center networks on commodity switches and servers have been studied for decades, and various novel network architectures have been proposed to tackle challenges of typical architectures [47]. The proposed designs fall into switch-centric [3,5,15,36,38] and server-centric [9,18,19,26,32] architectures in terms of forwarding intelligence, or fall into rigid [3,15,29,36] and flexible [4,10,22,37,38,44] architectures in the light of characteristics of topologies. In this section, we introduces some of the major architectures with respect to rigid and flexible structures.

Multi-rooted tree is widely used in data center network design [3,15,23,36], which provides large bandwidth and efficient routing protocols on homogeneous switches. Fat-tree [3] provides full bisection bandwidth and a large number of parallel paths between servers. VL2 [15] focuses on the issue of performance isolation, flat addressing, and load balance. Portland [36] studies migration of virtual machines on multi-rooted tree based networks. Monsoon [16] creates a mesh switching domain by using programmable commodity layer-2 switches, where all servers are connected by a layer 2 network. Recursive mechanisms are used in server-centric network designs, where the scale of network is determined by the switch ports count and the number of levels. The typical architectures include DCell [19], BCube [18], FiConn [26] and HCN [20]. MDCube [42] focuses on the interconnection among modular data centers. CamCube [1,9], DPillar [30], SWCube [27], and FSquare [28] are constructed on the 3D Torus, wrapped butterfly, generalized hypercube,and compound Clos structures, respectively. The above architectures focus on the high performance on homogeneous switches, and pay less attention to design flexibility of networks. They support network scale with a coarse parameter, which is completely determined either by the number of ports in a switch, or by both the number of levels and switch ports.

Several flexible solutions have been proposed to support design flexibility in network construction. Scafida [22] and FScafida [21] provides small increment by randomness interconnection on asymmetric data center network. LEGUP [12] preserves available ports for future expansion on Clos networks. REWIRE [11] focuses on construction of network that maximizes performance with a given budget. Small world data centers [37] are constructed by a hybrid method of regular and random connection, which supports the greediest routing protocols. SecondNet [17] is a virtualization architecture, which focuses on protocol scalability and resource utilization of the infrastructure network other than the structure itself. Jellyfish [38] deploys random connection among switches to support arbitrary-size network and incremental growth. Jellyfish can support switch heterogeneity, while the random connection can not optimize network performance with the identical equipments. PAST [39] provides a multi-path solution at the cost of degrading throughput in Jellyfish. S2 deploys multiple rings and random connection to construct data center networks, where the ring connection is coordinates-based. S2 can provide rich bandwidth in topology and scalability in protocols. Most of the current flexible networks introduce random connection to improve the flexibility of architecture, as in Scafida [22], SWDC [37], Jellyfish [38], and S2 [44,45] . The random connection cannot always provide high network performance, which maybe result in a large network diameter or critical link in random connection.

Our work differs from the current flexible proposals in three key aspects. First, the proposed algorithm provides a higher performance on identical equipments. Second, the proposed algorithm supports incremental deployment, as well as asymmetrical inter-switch ports count. Third, the proposed Length Shuffle connection can be applied in other flexible architectures to reduce the average length of routing path.

3. Length Shuffle connection

In this section, we introduce Length Shuffle connection, which creates a graph on arbitrary number of nodes with degree constraint of each nodes. Given nodes with degree constraint, Length Shuffle connection targets for flexible graph with a small number of long path among nodes.

3.1. Algorithm of Length Shuffle

Let N denote the number of nodes, and each node is assigned a unique ID v_i, taking a value from [1,N]. Node v_i has a degree of d_i, which can connect to at most d_i ($d_i \geq 2$) other nodes. Length Shuffle connects all these nodes into a graph with their degree constraints, which follows a greedy principle that connects the furthest two possible nodes gradually. The distance between any two nodes is defined as the length of the shortest path between them on the current graph of all nodes. In the initialization stage, there is no connection between any two nodes in the graph, and the distance between any two nodes is infinite.

Let S_v denote the set of nodes which has at least one available degree left on current stage. Let $Dis(v_a, v_b)$ denote the length between v_a and v_b, where $v_a \in S_v$ and $v_b \in S_v$. Let $MaxDis$ denote the maximum distance between any two nodes in S_v of the current graph. Let S_c denote a set of node pairs, where each element (v_i, v_j) contains two nodes $v_i \in S_v$ and $v_j \in S_v$ with $Dis(v_i, v_j) = MaxDis$. The procedure of Length Shuffle connection is shown in Algorithm 1. The procedure first gets the set of S_v, where each node

Algorithm 1 Length Shuffle connection.

1: get set S_v;
2: update $Dis(v_a, v_b)$ where $v_a \in S_v$ and $v_b \in S_v$;
3: obtain set S_c where $Dis(v_i, v_j) = MaxDis$;
4: randomly select (v'_i, v'_j) in S_c;
5: connect v'_i and v'_j;
6: update degree state of v'_i and v'_j;
7: repeat the step 1–6 until $|S_v| = 0$ or $|S_v| = 1$;